ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY (AAiT)

School of Information Technology and Engineering(SiTE)

MACHINE LEARNING AND BIG DATA

PROJECT ONE (LINEAR REGRESSION)

NAME: Natan Amanuel Mamo

Id no: ATR/7142/13

Year: IV (Software stream)

Submitted to: Mr Bisrat

TABLE OF CONTENT

Introduction to the Linear Regression Model for Churn Prediction

In the landscape of customer-centric businesses, understanding customer churn, where customers discontinue their relationship with a company, has become increasingly vital. This report focuses on the application of a linear regression model to predict churn rates within a given dataset of customer information.

This model utilizes multiple features from the dataset, such as demographic attributes (e.g., age, gender), financial indicators (e.g., credit score, balance, and salary), and account-specific metrics (e.g., tenure and number of products held). By analyzing the relationships between these variables and the dependent variable, exited (binary outcome), it aims to predict whether the customer has exited or not.

Linear regression is a powerful statistical method used for modeling the relationship between a dependent variable and one or more independent variables. The core principle of the linear regression algorithm is to find the best-fitting linear equation that describes the data points in a given dataset. By minimizing the difference between the predicted values and the actual observations, linear regression enables analysts to make informed predictions about the dependent variable based on the values of the independent variables. In contexts such as credit scoring or customer satisfaction assessment, as highlighted in datasets containing customer demographics and financial details, linear regression can effectively identify trends and relationships, providing valuable insights into factors influencing customer behavior and outcomes.

Although many of the models related to customer churn prediction use other algorithms like logistic regression, Linear regression is also suited for this analysis due to its simplicity and interpretability, allowing us to quantify the impact of various factors on customer churn. The coefficients obtained through this model reveal which factors most significantly influence a customer's decision to stay or leave, thus informing strategies for customer retention. This report tries to describe the methodologies and tools used, model training, evaluation metrics employed, and the implications of the results obtained through the analysis.

ABOUT THE DATASET

The dataset used for this analysis was sourced from Kaggle, a well-known platform for data science and machine learning resources. This comprehensive dataset includes information on customer demographics and financial behavior from a bank, comprising 20 attributes.

Attributes:

- CustomerId**: A unique identifier for each customer.
- Surname: The last name of the customer.
- CreditScore: A numerical score representing the customer's creditworthiness.
- Geography: The country of residence for the customer.
- Gender: gender, which can provide insight into demographic trends.
- Age: The age of the customer, which can be correlated with financial behavior.
- Tenure: The number of years the customer has been with the bank.
- Balance: The customer's account balance, indicating their financial status.
- NumOfProducts: The number of financial products the customer has with the bank.
- HasCrCard: A binary indicator of whether the customer has a credit card.
- IsActiveMember: A binary indicator of whether the customer is an active member.
- EstimatedSalary: The estimated salary of the customer.
- Exited: A binary variable indicating whether the customer has left the bank.
- Complain: A binary variable indicating whether the customer has filed a complaint.
- Satisfaction Score: A score reflecting the customer’s satisfaction with the bank.
- Card Type: The type of credit card held by the customer.
- Point Earned: The loyalty points earned by the customer through banking activities.

This dataset is particularly valuable for predictive modeling and customer segmentation, allowing for the identification of patterns and trends.

Jupyter Notebook

I used Jupyter Notebook for my data analysis tasks in Python. Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It is widely used in data science and machine learning for its ability to support interactive data visualization and the integration of code, which facilitates exploratory data analysis and prototyping.

CODE BREAKDOWN

IMPORTING LIBRARIES

In this project, I imported several essential libraries to facilitate data manipulation and analysis. Below is a breakdown of each imported library:

- Pandas(`import pandas as pd`): Pandas is a powerful data manipulation library that provides flexible data structures, like Series and DataFrames, for complex data analysis tasks. It is widely used for reading, writing, and processing structured data efficiently, enabling operations like filtering, grouping, and aggregating.
- NumPy(`import numpy as np`): NumPy is a foundational library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, alongside a collection of mathematical functions to operate on these arrays. It excels in performing efficient numerical operations and handling linear algebra.
- Matplotlib(`import matplotlib.pyplot as plt`): Matplotlib is a plotting library that is used for creating static, interactive, and animated visualizations in Python. It provides functionalities to create various types of plots and charts, enabling data visualization to help interpret data better.
- Seaborn(`import seaborn as sns`): Seaborn is built on top of Matplotlib and simplifies the creation of attractive statistical graphics. It helps visualize complex datasets with simpler interfaces and provides enhanced visual appeal by improving the aesthetics of visualizations.
- Scikit-learn (`from sklearn.model_selection import train_test_split`): Scikit-learn is a comprehensive library for machine learning that offers simple and efficient tools for data mining and data analysis. The module imported is used for splitting datasets into training and testing sets, which is essential in evaluating the performance of machine learning models.
- Scipy(`import scipy`): SciPy is a scientific computing library that builds on NumPy and provides additional functionality such as optimization, integration, interpolation, eigenvalue problems, and more. It is widely used in mathematical computations and scientific research.
- Statsmodels(`import statsmodels.api as sm`): Statsmodels is a library for estimating and testing various statistical models. It offers classes and functions for performing statistical tests, estimating statistical models, and visualizing results, making it a favorite for statistical analysis in Python.

Each of these libraries plays a crucial role in the workflow of data analysis, ensuring the efficient manipulation, analysis, and visualization of the dataset.

READ FROM FILE

Let's break down the given code for reading a CSV file and managing its data in Python using the Pandas library.

1. Reading the CSV File (df = pd.read_csv('Customer-Churn-Records.csv')): This function from the Pandas library is used to read a CSV (Comma Separated Values) file. The resulting data is stored in a DataFrame called **DF**. A DataFrame is a 2-dimensional labeled data structure with columns that can be of different types (similar to a table in a database or a spreadsheet).

2. Viewing the First Few Rows (df.head()): This function displays the first five rows of the DataFrame by default. It allows you to quickly glance at the data to understand its structure and content.

3. Getting Basic Information About the DataFrame (`df.info()`): This function provides a concise summary of the DataFrame, including the number of entries, the data types of each column, and the number of non-null values. This is helpful for understanding the structure of the data and identifying potential data quality issues.

4. Checking for Missing Values: `df.isnull()` returns a DataFrame of the same shape as `df`, containing `True` for each missing value and `False` otherwise. **`.sum()`** function aggregates the results of the previous step, returning a Series with the total count of missing values for each column. This is useful for identifying columns that may require data cleaning or imputation.

5. Descriptive Statistics of the DataFrame: `df.describe()` This function generates descriptive statistics of numerical columns, including count, mean, standard deviation, min, max, and quartiles. It's useful for getting a statistical summary of the data and understanding the distribution of numeric variables.

ADJUST THE DATASET

```
df.drop(columns=['RowNumber', 'CustomerId', 'Surname'], inplace=True)

df = pd.get_dummies(df, drop_first=True)

X = df.drop('Exited', axis=1)  # Features

y = df['Exited']
```

1. Dropping Columns: Remove specific columns that may not contribute useful information for analysis or predictive modeling. Columns Dropped: **`RowNumber`, `CustomerId`, `Surname`**. Use of **`inplace=True`** Changes are applied directly to `df` without needing to assign it back to another variable.

2. Creating Dummy Variables: Convert categorical variables into a numerical format. This conversion allows algorithms to analyze the data effectively since they typically work with numerical types.

SEPARATING FEATURE VARIABLES AND TARGET VARIABLE

Separating Features and Target: Features (`X`), All columns in `df` except the target variable **`Exited`**, which includes various financial and demographic details pertinent for predictive analysis. **Target (`y`),** The `Exited` column represents the outcome variable that the model will try to predict, likely indicating whether a customer has exited the service or product.

<u>DIVIDE DATA (TRAINING AND TEST SET)</u>

The `train_test_split` function from the `sklearn.model_selection` library to split a dataset into training and testing subsets.

1. Function Used: `train_test_split()` is a function that randomly splits arrays or matrices into two subsets: one for training a model and the other for testing it.

2. Variables:

- `X_train`: This variable contains the feature data that will be used to train the model.
- `X_test`: This variable contains the feature data that will be used to test the model.
- `y_train`: This variable contains the target labels corresponding to `X_train`.
- `y_test`: This variable contains the target labels corresponding to `X_test`.

`test_size=0.2`: This specifies that 20% of the data should be held back for testing, while the remaining 80% will be used for training the model.

`random_state=42`: This is a seed for the random number generator, ensuring that the split is reproducible. Every time the code runs, the same split of data will result if random_state is set to the same value.

By splitting the dataset into `X_train`, `X_test`, `y_train`, and `y_test`, the code prepares the data for training a predictive model, in our case, predicting customer churn. The training subset (`X_train`, `y_train`) will be used to train the model, while the testing subset (`X_test`, `y_test`) will help evaluate how well the model generalizes to new, unseen data.

<u>CREATE AND TRAIN MODEL</u>

1. `model = LinearRegression()`: This line imports a class named `LinearRegression` from scikit-learn's linear model module and creates an instance of it. The `model` variable now holds this instance, which will be used to perform linear regression.

2. `model.fit(X_train, y_train)`: The `fit` method is called on the `model` instance. This method takes two arguments: `X_train` and `y_train`. It uses the training data to learn the relationship between the features and the target variable by estimating the coefficients of the linear equation.

3. `y_pred = model.predict(X_test)`: The `predict` method of the `model` instance is called with `X_test`, which contains the features for which we want to make predictions. It uses the model that was trained in the previous step to generate predictions (`y_pred`) for the target variable based on the unseen test data. The output `y_pred` will contain the predicted values of the target variable corresponding to the input features in `X_test`.

<u>EVALUATION</u>

mse = mean_squared_error(y_test, y_pred): This refers to a function that computes the mean squared error (MSE) between two sets of data. It is commonly used to evaluate the performance of regression models by measuring the average squared difference between actual and predicted values.

r2 = r2_score(y_test, y_pred): This refers to a function that calculates the R^2 (R-squared) score. R^2 is a statistical measure that represents the proportion of variance for a dependent variable that's explained by independent variables in a regression model.

The first line calculates the mean squared error of the model's predictions compared to the actual test values, providing a single numerical evaluation of the prediction accuracy.

The second line calculates the R^2 score, which serves as another measure of how well the model performs, indicating the proportion of variance explained by the model.

VISUALIZE THE MODEL

1. `plt.scatter(y_test, y_pred)`: This line creates a scatter plot with the true target values (`y_test`) on the x-axis and the predicted values (`y_pred`) on the y-axis. It visually represents how well the model's predictions match the actual target values.

2. `coefficients = pd.DataFrame(model.coef_, X.columns, columns=['Coefficient'])`: This line creates a DataFrame from the coefficients of a model (`model.coef_`), using the feature names from `X.columns` as the index and labeling the column as "Coefficient". This helps in understanding the importance of each feature in the model.

3. `plt.scatter(y_pred, residuals)`: This line creates another scatter plot, this time plotting predicted values (`y_pred`) on the x-axis and the computed residuals on the y-axis. This helps to visualize the distribution of residuals and can indicate if there's a pattern (which would suggest problems with the model).

4. `sns.histplot(residuals, kde=True)`: This line creates a histogram of the residuals using Seaborn, with `kde=True` adding a Kernel Density Estimate (KDE) line to visualize the distribution of residuals more smoothly.

6. `stats.probplot(residuals, plot=plt)`: This line generates a Q-Q plot using the residuals, plotting them against a theoretical normal distribution to assess if the residuals are normally distributed. The `plot=plt` argument means that it will be drawn on the current Matplotlib plot.

This part of the code generates several visualizations and analyses related to the performance of a predictive model. It compares actual vs. predicted values, analyzes residuals, checks their distribution, and assesses normality through the Q-Q plot, providing insights into the model's effectiveness.

MODIFY HYPERPARAMETERS

The provided code snippet uses the Scikit-learn library to implement two types of linear regression models: Ridge regression and Lasso regression. Here’s a detailed breakdown of what the code does:

The models created (`ridge_model` and `lasso_model`) would be used to predict the likelihood of customers exiting based on their attributes (features represented in `X_train`).

Ridge regression would be used when all features are believed to contribute to the output, while Lasso regression would be beneficial when we suspect that only a subset of those features is important.

In summary, the code builds and fits two regression models to analyze the relationship between numerous input features (from the customer dataset) and the target variable (customer retention or exit status). This could be part of a larger analysis to find the most significant factors influencing customer behavior.

EVALUATION OF THE MODEL

Evaluation Metrics: Mean Squared Error (MSE) and **R-squared** are two common metrics used to evaluate the performance of a linear regression model.

Mean Squared Error (MSE):

- Measures the average squared difference between the predicted values and the actual values.
- A lower MSE indicates that the model's predictions are closer to the actual values.
- An MSE of 0.0009991648835770575 is quite low, suggesting that the model is making accurate predictions.

R-squared:

- Represents the proportion of the variance in the dependent variable that is explained by the independent variables.
- A higher R-squared value indicates a better fit of the model to the data.
- R-squared of 0.9936716757089954 is very high, meaning that 99.36% of the variance in the dependent variable is explained by the model.

Overall evaluation of the model:

Depending on the above two evaluation metrics, this linear regression model is performing exceptionally given that their values are very close to the ideal values that a good linear regression algorithm produces. It's making highly accurate predictions and capturing a large portion of the variability in the data.

VISUAL REPRESENTATIONS (PLOTS)

The Scatter Plot

The scatter plot provides a qualitative understanding of the model's predictions. The code generates a scatter plot that visualizes the relationship between the actual values (y-axis) and the predicted values (x-axis) for the linear regression model.

In the model, the data points are concentrated in four corners of the plot. When the data points are concentrated in four corners of the plot, it suggests a potential issue with the model's predictions. It suggests that the model might not be capturing the full range of the data or might be making systematic errors.

The histogram

The histogram shows the distribution of residuals, which are the differences between the actual values and the predicted values from the model.

Here's what the plot indicates:

1. Normal Distribution: The histogram appears to be roughly bell-shaped, suggesting that the residuals are normally distributed. This is a good indication that the model's assumptions are met.

2. Centered around Zero: The majority of the residuals are clustered around zero. This means that, on average, the model's predictions are close to the actual values.

3. Variance: The spread of the residuals indicates the model's accuracy. A narrower distribution suggests that the model's predictions are more precise.

Overall, the histogram suggests that your linear regression model is performing well. The normally distributed residuals indicate that the model's assumptions are met, and the clustering around zero suggests accurate predictions.

The Q-Q Plot

It is a Q-Q (Quantile-Quantile) plot of the residuals from your linear regression model. It's used to assess whether the residuals are normally distributed.

- Normal Distribution: If the residuals are normally distributed, the points on the plot should roughly follow a straight line.
- Deviations from Normality**:** Deviations from the straight line indicate non-normality.

Q-Q plot of this model shows that the residuals are nearly normally distributed, even though some points deviate significantly from the straight line, especially in the tails.

Modifying Hyperparameters for Linear Regression

While linear regression doesn't have many traditional hyperparameters like other complex models, we can still influence its performance by adjusting certain parameters related to the optimization process.

1. Regularization:

L1 Regularization (Lasso Regression): This adds a penalty term to the loss function, encouraging the model to shrink coefficients of less important features towards zero. It can be useful for feature selection.

You can control the strength of regularization using the alpha hyperparameter. Higher values of alpha lead to stronger regularization.

2. Optimization Algorithm:

While linear regression is typically solved using ordinary least squares (OLS), you can experiment with different optimization algorithms, such as gradient descent, to see if they improve performance. However, for linear regression, OLS is generally the most efficient method.

```
Lasso(alpha=0.1)
```

This output indicates that you've created a Lasso regression model with an alpha value of 0.1.

Lasso Regression is a linear regression model that incorporates L1 regularization. This regularization technique adds a penalty term to the loss function, which encourages the model to shrink the coefficients of less important features towards zero. As a result, Lasso regression can perform feature selection automatically.

The alpha parameter controls the strength of the regularization. A higher alpha value leads to stronger regularization, which can result in more feature elimination. A lower alpha value allows for more complex models with more features.

In this case, alpha=0.1 means that a moderate level of regularization is applied. The model will balance the trade-off between fitting the training data well and avoiding overfitting.

CONCLUSION

In conclusion, this model has provided a comprehensive analysis of our customer dataset, highlighting critical insights into customer behavior, demographics, and satisfaction levels. Through the exploration of variables such as credit scores, geography, age, and account tenure, we have identified distinct patterns that could inform targeted marketing strategies and enhance customer engagement. Although linear regression is not seen as ideal algorithm for predicting binary target value, this model shows that linear regression can give accurate predictions.